CLAIMS

1. A method for testing software comprising:

modeling software using a software model that describes behavior associated with the software; and

operating on the software model using a random destination algorithm and at least one other different algorithm to produce a sequence of test actions, the random destination algorithm being configured to randomly select a destination in the model and move to that destination to produce the sequence of test actions.

2. The method of claim 1, wherein the software model comprises a state graph having multiple nodes individual ones of which representing a state, and links between the nodes that represent actions.

3. The method of claim 2, wherein said operating comprises using the random destination algorithm to select a destination node at random, independent of a present node, and traverse state space to arrive at the destination node.

4. The method of claim 2, wherein said operating comprises using the random destination algorithm to select a destination node at random, independent of any previously-traversed nodes, and traverse state space to arrive at the destination node.

5. The method of claim 2, wherein said operating comprises using the random destination algorithm to select a destination node at random, independent of a nearest neighbor node, and traverse state space to arrive at the destination node.

6. The method of claim 2, wherein the software model comprises clusters of related nodes, and said operating comprises using the random destination algorithm to select, at random, at least one cluster of nodes.

7. The method of claim 2, wherein the software model comprises clusters of related nodes, and said operating comprises using the random destination algorithm to select, at random, at least one node inside at least one cluster of nodes.

8. One or more computer-readable media having computer-readable instructions thereon which, when executed by a computer, cause the computer to:

model software using a software model that describes behavior associated with the software, the software model comprising a state graph having multiple nodes individual ones of which represent a state, and links between the nodes that represent actions; and

operate on the software model using a random destination algorithm and at least one other different algorithm to produce a sequence of test actions, the random destination algorithm being configured to randomly select a destination node in the model and move to that destination node to produce the sequence of

test actions, the selection of the destination node being performed independent of any previously-traversed nodes, and independent of any nearest neighbor nodes.

9. A method of testing software comprising:

modeling software using a software model that describes behavior associated with the software;

operating on the software model using a random destination algorithm to produce a sequence of test actions, the random destination algorithm being configured to randomly select a destination in the model and move to that destination to produce the sequence of test actions; and

operating on the software model using multiple other algorithms that are different from the random destination algorithm to produce a further sequence of test actions.

10. The method of claim 9, wherein said modeling comprises using a state graph having multiple nodes individual ones of which represent a state, and links between the nodes that represent actions.

11. The method of claim 9, wherein said multiple other algorithms comprise a random walk algorithm.

12. The method of claim 9, wherein said multiple other algorithms comprise a Chinese postman algorithm.

13. The method of claim 9, wherein said multiple other algorithms comprise a Markov chain algorithm.

14. The method of claim 9, wherein said multiple other algorithms comprise a anti-random walk algorithm.

15. The method of claim 9, wherein said multiple other algorithms comprise an algorithm selected from a group comprising: a random walk algorithm, a Chinese postman algorithm, a Markov chain algorithm, and a anti-random walk algorithm.

16. One or more computer-readable media having computer-readable instructions thereon which, when executed by a computer, cause the computer to:

operate on a software model using a random destination algorithm to produce a sequence of test actions, the software model comprising a state graph having multiple nodes individual ones of which represent a state, and links between the nodes that represent actions, the random destination algorithm being configured to randomly select a destination node in the state graph and move to that destination node to produce the sequence of test actions; and

operate on the software model using multiple other algorithms that are different from the random destination algorithm to produce a further sequence of test actions, the multiple other algorithms being selected from a group comprising: a random walk algorithm, a Chinese postman algorithm, a Markov chain algorithm, and a anti-random walk algorithm.

17. A method of testing software comprising:

traversing a state graph that models software, the state graph having multiple nodes individual ones of which represent a state, and links between the nodes that represent actions, said traversing using an algorithm having a first graph traversal characteristic to produce a sequence of test actions; and

traversing the state graph using an algorithm having a second graph traversal characteristic that is different from the first graph traversal characteristic to produce a further sequence of test actions.

18. The method of claim 17, wherein the algorithms are different.

19. The method of claim 17, wherein the algorithm having the first graph traversal characteristic is one selected from a group of algorithms comprising: a random walk algorithm, a random destination algorithm, and a anti-random walk algorithm.

20. The method of claim 19, wherein the algorithm having the second graph traversal characteristic different from the algorithm having the first graph traversal characteristic and is one selected from a group of algorithms comprising: a random walk algorithm, a random destination algorithm, and a anti-random walk algorithm.

21. One or more computer-readable media having computer-readable instructions thereon which, when executed by a computer, cause the computer to implement the method of claim 17.

22. A method of testing software comprising:

traversing a state graph using a deterministic first algorithm to produce a sequence of test actions, the state graph having multiple nodes individual ones of which represent a state, and links between the nodes that represent actions; and

traversing the state graph using a second algorithm that is less deterministic than the first algorithm to produce a further sequence of test actions.

23. A method of testing software comprising:

traversing a state graph using a random walk first algorithm to produce a sequence of test actions, the state graph having multiple nodes individual ones of which represent a state, and links between the nodes that represent actions; and

traversing the state graph using a second algorithm that is less random than the first algorithm to produce a further sequence of test actions.

24. A method of testing software comprising:

providing one or more algorithms for operating on a software model that describes behavior associated with software that is to be tested;

selecting one or more algorithms;

operating on the software model using the selected one or more algorithms to produce a sequence of test actions;

changing the selected one or more algorithms; and

operating on the software model using one or more changed algorithms.

25. The method of claim 24, wherein said changing comprises changing a way an algorithm interacts with the software model.

26. The method of claim 25, wherein said changing comprises changing one or more properties associated with an algorithm.

27. The method of claim 24, wherein said changing comprises selecting at least one different algorithm.

28. One or more computer-readable media having computer-readable instructions thereon which, when executed by a computer, cause the computer to:

provide one or more algorithms for operating on a software model that describes behavior associated with software that is to be tested;

select multiple algorithms to define a first collection of algorithms;

operate on the software model using the first collection of algorithms to produce a sequence of test actions;

change at least one of the selected algorithms to define a second collection of algorithms; and

operate on the software model using the second collection of algorithms to produce an additional sequence of test actions.

29. A method of testing software comprising:

traversing a state graph using a random destination algorithm, the state graph having multiple nodes individual ones of which representing a state, and links between the nodes that represent actions, said traversing producing a sequence of test actions; and

traversing the state graph using multiple steps from a random walk algorithm to produce an additional sequence of test actions.

30. The method of claim 29 further comprising traversing the state graph using a random destination algorithm after said traversing of the state graph using the random walk algorithm.

31. The method of claim 29, wherein said traversing using multiple steps comprises using a predetermined number of steps.

32. The method of claim 29, wherein said traversing using multiple steps comprises using a random number of steps.

33. The method of claim 29, wherein said acts of traversing comprise iterating through the random destination and random walk algorithms.

34. The method of claim 33, wherein said traversing using multiple steps comprises changing the number of steps on at least one iteration.

35. The method of claim 33, wherein said traversing using multiple steps comprises randomly changing the number of steps on at least one iteration.

36. The method of claim 33, wherein said traversing using multiple steps comprises changing the number of steps on at least one iteration in accordance with probabilistic characteristics.

37. A method of testing software comprising:

selecting a first algorithm from among a number of different algorithms;

operating on a software model that describes behavior of software that is to be tested, said operating taking N steps using the first algorithm, where N is an integer and said steps produce a sequence of test actions;

selecting a second algorithm from among the number of different algorithms, the second algorithm being different from the first algorithm; and

operating on the software model by taking N1 steps using the second algorithm, where N1 is an integer, said N1 steps producing an additional sequence of test actions.

38. The method of claim 37, wherein the algorithms are different based upon how they interact with the software model.

39. The method of claim 37, wherein the algorithms are different based upon how they traverse a graph that describes the software's behavior.

40. The method of claim 37, wherein said number of different algorithms include at least one algorithm that is more deterministic than at least one other algorithm.

41. The method of claim 37, wherein said number of different algorithms include at least one algorithm that is more random than at least one other algorithm.

42. The method of claim 37, wherein at least one of said acts of selecting is based on the structure of the software model.

43. The method of claim 37, wherein N and N1 are predetermined.

44. The method of claim 37, wherein N and N1 are preprogrammed.

45. The method of claim 37, wherein N and N1 are randomly selected.

46. The method of claim 37, wherein N and N1 are calculated using a Poisson distribution having multiple values each with an assigned probability of being selected.

47. The method of claim 46, wherein one or more assigned probabilities change over time.

48. The method of claim 37 further comprising iterating through said acts of operating at least one time.

49. The method of claim 48 further comprising changing the values of N and N1 during the iteration.

50. The method of claim 48 further comprising:

assigning values to N and N1 using a first method on a first pass; and

assigning values to N and N1 using a second method that is different from the first method on a second pass.

51. The method of claim 37 further comprising replacing one or more of the algorithms used to operate on the software model after a certain period of time.

52. The method of claim 37 further comprising replacing one or more of the algorithms used to operate on the software model after the one or more algorithms have been used a certain number of times.

53. A method of testing software comprising:

representing software using a model that describes the software's behavior, the software having an associated social context; and

selecting one or more algorithms to operate upon the model as a function of the software's social context; and

operating upon the model using the selected one or more algorithms to produce a sequence of test actions.

54. The method of claim 53, wherein the social context is associated with a software developer who developed the software.

55. The method of claim 53 further comprising:

changing the one or more algorithms; and

operating upon the model using changed algorithms to produce an additional sequence of test actions.

56. A method of testing software comprising:

developing a profile associated with one or more software developers, the profile describing one or more algorithms that are more likely to identify problems associated with software developed by the one or more software developers;

selecting, from a developer's profile, one or more algorithms when a software model associated with the developer's software is to be operated upon; and

operating upon the software model using the selected one or more algorithms to produce a sequence of test actions.

57. A method of testing software comprising:

defining one or more clusters in a software model that models software that is to be tested;

providing multiple different algorithms for operating upon the software model;

selecting a first algorithm for operating on the software model to produce a sequence of test actions;

selecting a second algorithm that is different from the first algorithm for operating on the software model to produce an additional sequence of test actions; and

operating on the software model using the first and second algorithms to produce the sequences of test actions, one of the first and second algorithms having a better chance at accessing a cluster than the other of the first and second algorithms.

58. The method of claim 57, wherein said software model comprises a state graph having multiple nodes and links between the nodes, individual nodes representing states, individual links representing actions that move between states.

59. The method of claim 58, wherein the first and second algorithms have different graph traversal characteristics.

60. The method of claim 58, wherein said defining comprises defining the clusters based on areas of connectivity within the state graph.

61. The method of claim 57, wherein said defining comprises defining the clusters based on the structure of the software.

62. The method of claim 57, wherein one of said first and second algorithms comprises a random destination algorithm.

63. The method of claim 57, wherein one of said first and second algorithms comprises a random walk algorithm.

64. The method of claim 57, wherein one of said first and second algorithms comprises a anti-random walk algorithm.

65. The method of claim 57, wherein one of said first and second algorithms comprises a deterministic algorithm, and the other algorithm comprises a non-deterministic algorithm.

66. A software-testing system comprising:

a software model processor configured to:

receive a software model that describes behavior associated with software that is to be tested, and

operate upon the model to provide a sequence of test commands for testing the software; and

an algorithm set associated with the model processor and comprising multiple different algorithms, the software model processor being configured to select at least two different algorithms and use the algorithms to operate upon the software model to produce the sequence of test commands.

67. The software-testing system of claim 66, wherein the model processor is configured to change one or more of the algorithms.

68. The software-testing system of claim 66, wherein at least one algorithm comprises a random walk algorithm.

69. The software-testing system of claim 66, wherein at least one algorithm comprises a Chinese Postman algorithm.

70. The software-testing system of claim 66, wherein at least one algorithm comprises a Markov chain algorithm.

71. The software-testing system of claim 66, wherein at least one algorithm comprises a anti-random walk algorithm.

72. The software-testing system of claim 66, wherein at least one algorithm is selected from a group comprising a random walk algorithm, a Chinese Postman algorithm, a Markov chain algorithm, and a anti-random walk algorithm.

73. A software-testing system comprising:

a software model processor configured to:

receive a software model in the form of a state graph that describes behavior associated with software, the state graph having multiple nodes that represent state, and links between the nodes that represent actions, and

traverse the state graph to provide a sequence of commands for testing the software;

an algorithm set associated with the model processor and comprising multiple different algorithms; and

a graph traverser associated with the model processor and configured to:

traverse the state graph using an algorithm from the algorithm set, the algorithm having a first graph traversal characteristic to produce a sequence of test commands, and

traverse graph with an algorithm from the algorithm set having a second graph traversal characteristic that is different from the first graph traversal characteristic to produce a further sequence of test commands.

74. The software-testing system of claim 73, wherein the algorithm having the first graph traversal characteristic is selected from a group of algorithms comprising: random walk algorithms, random destination algorithms, and anti-random walk algorithms.

75. The software-testing system of claim 74, wherein the algorithm having the second graph traversal characteristic is selected from a group of algorithms comprising: random walk algorithms, random destination algorithms, and anti-random walk algorithms.

76. A software-testing system comprising:

means for receiving a software model;

means for operating on the software model in a first manner to produce a sequence of test actions; and

means for operating on the software model in different additional manners to produce additional sequences of test actions.

77. The software-testing system of claim 76, wherein said means for operating comprises multiple different graph traversal algorithms.

78. A method of modeling user behavior comprising:

representing software using a model comprising a state graph, the state graph having multiple nodes individual ones of which represent a state, and links between the nodes that represent actions;

traversing the state graph using an algorithm having a first graph traversal characteristic to produce a sequence of user actions; and

traversing the state graph using an algorithm having a second graph traversal characteristic that is different from the first graph traversal characteristic to produce a further sequence of user actions.